**Assignment 5:** Begin a transaction, perform a series of INSERTs into 'orders', setting a SAVEPOINT after each, rollback to the second SAVEPOINT, and COMMIT the overall transaction.

Let's break down the task into steps involving transactions, SAVEPOINTs, and rollbacks in SQL:

**SQL Statements**

**1. BEGIN Transaction:** Start a new transaction.

BEGIN TRANSACTION;

**2. INSAERT Into `order` Table (First INSERT):** Insert the first record into the orders table and set a SAVEPOINT.

SAVEPOINT savepoint1;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_amount)

VALUES (nextval('orders\_order\_id\_seq'), 123, CURRENT\_DATE, 100.00);

- This assumes you are using a sequence (**`orders\_order\_id\_seq`**) for generating **`order\_id`**, and **`123`** is a placeholder for **`customer\_id`** . **`CURRENT\_DATE`** is used to insert the current date as **`order\_date`**, and **`100.00`** is a placeholder for **`order\_amount`**.

**3. INSERT Into `orders` Table (Second INSERT):** Insert the second record into the **`orders`** table and set another SAVEPOINT.

SAVEPOINT savepoint2;

INSERT INTO orders (order\_id, customer\_id, order\_date, order\_amount)

VALUES (nextval('orders\_order\_id\_seq'), 456, CURRENT\_DATE, 150.00);

**-** Adjust **`456`** and **`150.00`** with actual values for **`customer\_id`** and **`order\_amount`**.

**4. ROLLBACK to the Second SAVEPOINT:** Rollback to the second SAVEPOINT, undoing the second INSERT operation.

ROLLBACK TO SAVEPOINT savepoint2;

**5. COMMIT Transaction:** Commit the overall transaction to make the first INSERT permanent and discard the changes after the second SAVEPOINT rollback.

COMMIT;

**Explanation**

**- BEGIN TRANSACTION:** Starts a new transaction. All subsequent operations (inserts and rollbacks) are part of this transaction until it is either committed or rolled back completely.

**- SAVEPOINT:** Creates a point in the transaction to which you can roll back later if needed. Here, we set **`savepoint1`** after the first INSERT and **`savepoint2`** after the second INSERT.

**- INSERT INTO `orders`:** Inserts records into the `orders` table. Adjust the values (**`customer\_id`, `order\_date`, `order\_amount`**) according to your schema and requirements.

**- ROLLBACK TO SAVEPOINT:** Rolls back the transaction to the specified SAVEPOINT (**`savepoint2`** in this case), undoing the changes made after that SAVEPOINT.

**- COMMIT:** Commits the transaction, making all changes up to the first SAVEPOINT permanent in the database. Changes after the second SAVEPOINT are rolled back.

**Notes**

**-** SAVEPOINTs allow for partial rollback within a transaction, enabling finer control over undoing changes.

**-** Ensure your database system supports transactions (**`BEGIN`, `COMMIT`, `ROLLBACK`**) and SAVEPOINTs.

**-** Replace placeholder values (**`123`, `456`, `100.00`**, etc.) with actual data relevant to your application and database schema.

**-** This approach is useful when you want to ensure some parts of a transaction are committed while others are rolled back based on specific conditions or requirements.